

Data Stream Analysis and Data Mining With Storm

Gregor Majcen, Miha Zidar

Abstract—Twitter Storm is a powerful distributed real time data processing solution, with a wide range of usage. In this paper, we are going to take a look at how we can utilize Twitter Storms power for data mining on streams of data. The main focus of this paper is real time data processing and online data mining.

Index Terms—online learning, continuous data, data mining, distributed systems, horizontal scaling, batch processing

I. INTRODUCTION

TODAY we are generating more data per second than ever before, and the amount of data produced is only increasing over time. Data on its own is not that useful for us, unless we can extract information from it and the speed of gathering that information is becoming more and more valuable. This is where the real time data processing comes in. Big companies like Twitter, Groupon, spider.io and others, are using Twitter Storm to provide a better user experience.

In the last few years data processing has come a long way with services like MapReduce, Amazon EMR, Hadoop, and related technologies. All of these were made to handle massive amounts of data, and they do that very effectively. But lately their weakness is showing in their lack of real time processing. Now speed is starting to be ever more important, to get ahead of competition. This is where Storm comes in. Unlike other solutions that work in "batch" mode, Storm is made so it can handle data streams. With that it's possible to get calculations done on tuples of data and have the results back in matter of seconds, instead of hours or even days. One valuable ability to have, with ever increasing data streams, is scalability. To be more precise, horizontal scalability, where if the data stream grows, we can simply add more nodes without having to increase the speed of each individual node, and Twitter Storm provides that by running on top of Apache Zookeeper cluster.

Handling large data streams has always been possible even without Storm, but that meant handling queues and workers and dispatchers. Usually those present a problem with scaling, since dispatchers have to be modified to support new queues for the new workers. Thus a lot of the work with such system is just to keep the system running, instead of actually working on the logic of the workers. Storm keeps our focus on the actual logic of the topology, and it handles the queues and dispatchers so we do not have to.

We will begin by explaining what Storm is, how it works, and what it is used for. Then we will slowly narrow down all the Storm uses to the ones that become useful in data mining. We will show how a simple storm demo project, and discuss how these concepts can be applied to machine learning. At the end we will look at a practical machine learning algorithm

running on Storm, then we'll take that knowledge to hypothesize a solution for the stock market prediction.

January 14, 2013

II. TWITTER STORM

Storm is a distributed and fault-tolerant realtime computation system. It provides a high abstraction layer with which we can run complex computations on a cluster of computers. Storm provides users with a general framework for performing computations on data streams in real-time, similar to how Hadoop provides users with a framework for performing batch processing operations. Because it runs on top of Zookeeper and has a good messaging system using tuples of data, it provides a good alternative to managing your own cluster with queues and workers.

Storm can be used for stream processing, processing messages, updating databases, updating online machine learning models in real time. Other uses also include continuous computation, doing a continuous query on data streams and streaming out the results to users as they are computed, and for distributed RPC.

A. Storm structure

Before we can start anything, we need to learn Storm terminology. Even though Storm is often referred as Hadoop in real-time, the fact that it handles data streams and does not by itself store data or even finish processing, means that it can not be compared to it or any other batch method. That is why Storm does not have processes, but it calls those topologies, which unlike a process never finish.

The two main parts of topologies are spouts and bolts. Each spout and bolt runs in multiple instances called tasks², that are distributed on a storm cluster³. The number of tasks given to a bolt is determined in our program and is does not depend on the underlying structure of the Storm cluster. That makes adding and removing nodes from the cluster easy and requires no extra programming work.

- **spout** - this is basically the input for the entire system, it is the point (or multiple points) where the outside data streams are connected. The main task is to fetch data and convert it to tuples so that bolts can access it. For example, spout could read from a Kestrel queue, Twitter streaming API, directly read some sensor, or it could even scrape data from the web.
- **bolt** - a simple operating unit, that receives and processes data from spouts or bolts and possibly generates an output stream for other bolts. Bolts can be simple

functions that modify, filter, aggregate or join data, or talk to databases.

- **stream** - unbound sequence of tuples that is created and processed in a parallel using a distributed cluster. Streams are used to connect spouts and bolts.
- **topology** - a complete Storm structor similar to a never ending process. It is composed of spouts and bolts which are connected with data streams between them. 1

Storm topologies run on top of Storm cluster 3. A Storm cluster is made up of two kinds of nodes: the master node and the worker nodes. The master node runs a daemon called *Nimbus* which distributes the code to worker nodes around the cluster, assigns tasks to individual machines and monitors to failures. Every worker node runs a daemon called *Supervisor* that listens for work assigned to it by Nimbus and starts or stops worker processes to do the assigned task. The coordination between Nimbus and Supervisors is done through a Zookeeper cluster.

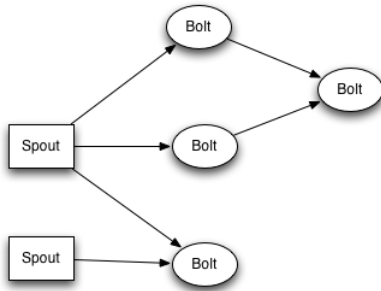


Fig. 1. Storm topology structor

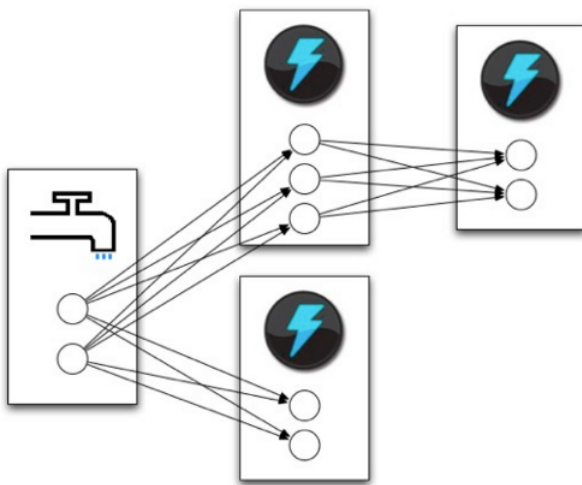


Fig. 2. Spouts and Bolts running as task and connected with streams.

B. Usage

- Stream processing -

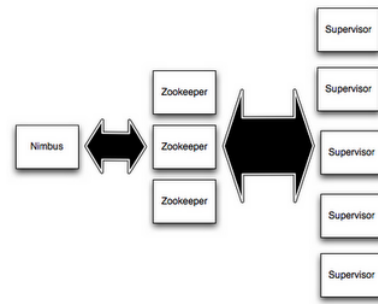


Fig. 3. Storm cluster

- Distributed RPC
- Continous computation

C. Simple Example

```
sample code:
do somethign
```

Explanation:

III. ONLINE MACHINE LEARNING

Online machine learning, is an induction model for learning from one instance at a time. This makes online algorithms perfect for data streams, since the model can be easily and constantly adjusted to acomodate for the newer data. As long as a good clasifier exists, the online algorithm will eventually learn to make better predictions. Online algorithms work with a sequence of trials, each of those can be represented with three steps. In the first step, the algorithm receives an instance, then in the second step, it predicts the label of the instance, and in the last step the algorithm receives the true label of the instance. The third step then also corrects the hypothesis to minimize the error, hopefully to make better predictions for future trials.

One drawback of these algorithms, is the need for more and more correctly labeled examples from which the algorithm would learn. This ofcourse is not possible, when correctly labeling an example costs money and time. Fortunately there are a lot of problems where the correct label is always available. For example any problem that has to do with predicting the future. In those problems the label will eventually be available and the algorithm will be able adjust its model to compensate the new data. A case of a future prediction problem would be movements on the stock market, or finding the next trending topic on twitter.

A few popular online machine learning algorithms are *perceptoron*, *winnow* and *naive bayes*.

A. Introduction

IV. MACHINE LEARNING WITH STORM

As we have seen in the previous sections, Storm is a very powerful tool, used for a lot of different purposes. In this

section, we will see implications Storm has in the data mining field.

A. preprocessing

Storm topologies can be used for machine learning indirectly, just for data preprocessing before running the data on a machine learning algorithm. The most common use for this would be video analyzing, image streams such as tumblr, or any other data type that needs preprocessing before any classification or regression can be done.

B. prediction

For this use case, Storm would act as distributed RPC that would just classify each instance from a given fixed model.

C. learning

V. CONCLUSION

ACKNOWLEDGEMENT

The authors would like to thank Matjaž Kukar, PhD Assistant Professor.

REFERENCES

- [1] authors *title* year.

Gregor Majcen 63070199



Miha Zidar 63060317

